To help Nginx recognize the web browsers and for telling the old from the modern, you need to insert multiple occurrences of the `ancient_browser` and `modern_browser` directives:

```
modern_browser opera 10.0;
```

With this example, if the User-Agent HTTP header contains Opera 10.0, the client browser is considered modern.

Map

Just like the Browser module, the Map module allows you to create maps of values depending on a variable:

```
map $uri $variable {
  /page.html  0;
  /contact.html  1;
  /index.html  2;
  default 0;
}
rewrite ^ /index.php?page=$variable;
```

Note that the `map` directive can only be inserted within the `http` block. Following this example, `$variable` may have three different values. If `$uri` was set to `/page.html`, `$variable` is now defined as `0`; if `$uri` was set to `/contact.html`, `$variable` is now `1`; if `$uri` was set to `/index.html`, `$variable` now equals `2`. For all other cases (`default`), `$variable` is set to `0`. The last instruction rewrites the URL accordingly. Apart from `default`, the `map` directive accepts another special keyword: `hostnames`. It allows you to match the hostnames using wildcards such as `*.domain.com`.

Two additional directives allow you to tweak the way Nginx manages the mechanism in memory:

- `map_hash_max_size`: Sets the maximum size of the hash table holding a map
- `map_hash_bucket_size`: Sets the maximum size of an entry in the map

Regular expressions may also be used in patterns if you prefix them with `~` (case sensitive) or `~*` (case insensitive):

```
map $http_referer $ref {
    ~google "Google";
    ~* yahoo "Yahoo";
    \~bing "Bing"; # not a regular expression due to the \ before the
tilde
    default $http_referer; # variables may be used
    }
```